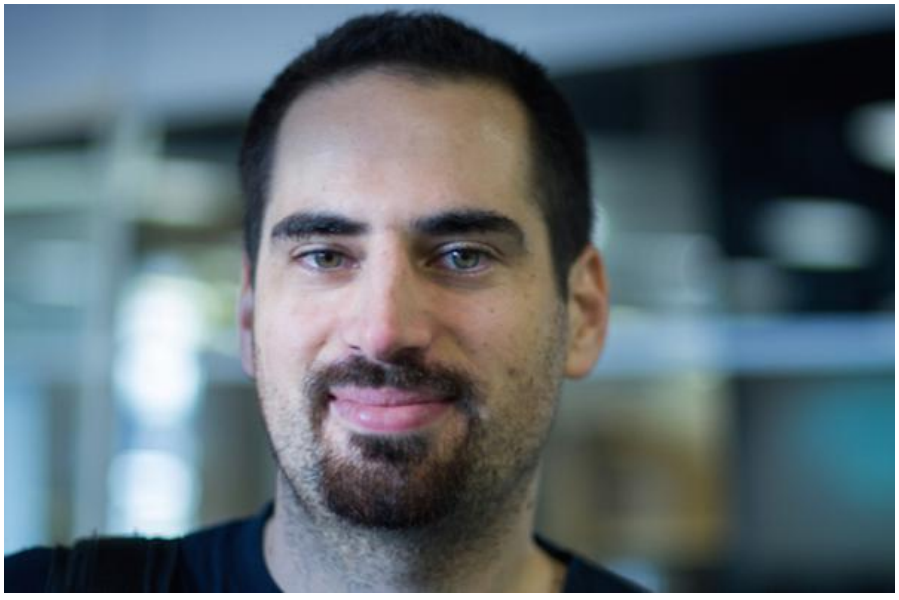
王小平的博客 分享
http://blog.sciencenet.cn/u/SciApple2014
关注计算机软件、人工智能和社会计算领域的创新，关注科学人文和社会文化的传播

博文

[转载]Olivier Grisel谈scikit-learn和机器学习技术的未来

已有 555 次阅读 2015-10-14 09:58 | 个人分类:新视角新观点 | 系统分类:观点评述 | 文章来源:转载

几周前，我们的Florian Douetteau (FD)对Olivier Grisel(OG)进行了一次访谈，正好我得到这个机会去旁听。Olivier是scikit-learn机器学习库的主要贡献者，因此他们两个详细地讨论了Olivier的工作和其它技术的发展。这是采访的第一部分。



Olivier Grisel 和 scikit-learn

FD：Olivier，你作为scikit-learn的主要贡献者已经有一段时间了。你可以告诉我们一些关于你的贡献么？
OG：大概是2010年，我就开始做scikit-learn这个项目。我是利用业余时间去做这个项目的。在2013年10月，我加入了Inria，一所面向计算机科学和自动化研究的法国研究院。我们有个团队，名叫Parietal，主要研究使用MRI数据对大脑进行建模。在这个项目中，我主要负责让scikit-learn发展地更长远，主要是指性能和可扩展性方面。
FD：scikit-learn已经发展了这么多年，而且知道开发过程中的许多阻碍。你能告诉我们一些关于将来的事么？
OG：当然可以。在过去的几年中，我们已经知道了一个重大的进展。现在，我们有很多新的用户和贡献者。根据我们的网站统计，我们每个月有150000到160000个独立访客，其中有1 / 3是回访用户，而且我们也有越来越多的贡献者。例如，在这些天，几乎有300个pull请求需要我们去处理。
scikit-learn大多数的新发展都来自用户社区自身的贡献。他们不断给scikit-learn库进行修改和补充，并为scikit-learn更好的后续版本提交这些工作。然后我们会对这些修改进行测试，并将其添加到每个新的版本中。例如，在最近的一个测试版本里，我们的一个贡献者开发了LDA估测器。这个算法在某种程度上可以替换scikit-learn已经存在的MMF，而且LDA在可扩展性方面表现的更强。

王小平
加为好友 给我留言
打个招呼 发送消息

扫一扫，分享此博文

作者的其他最新博文 全部

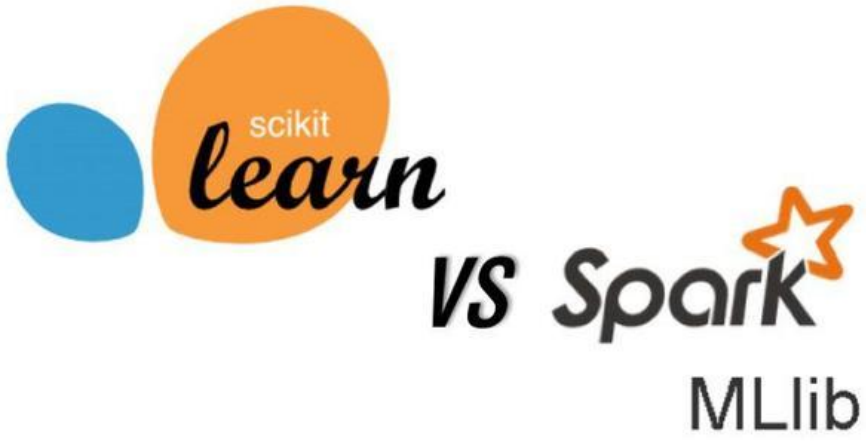
- [转载]SyncMind: 当大脑无法
- [转载]他是硅谷最好斗的华人
- [转载]ICML 2016丨「Data-
- [转载]清华人工智能论坛嘉宾
- [转载]谷歌大神 Jeff Dean 的
- [转载]谷歌神经网络如何实现

热门博文导读 全部

- 吓人的北美海棠果：卡片机...
- 你为什么必须被感动
- 究竟什么是基础研究？
- 《最忆是杭州》，最魅莫过...
- "你（工资）兑现后是多少？"
- 关于王贻芳教授评杨振宁教...

我开发的是一个更加长期的项目，这个项目涉及了大量的问题（因此它并不属于下一个版本的一部分）。我们正在努力使更多的scikit-learn算法能够以数据流模式，或核外模式，来管理数据，而不是在内存中控制整个数据集。我们希望它们逐渐地加载数据集，就像它们训练模型那样。

scikit-learn VS MLlib



FD：目前，在机器学习领域，我们听到了大量关于Spark的传闻。你有机会去尝试一下么？如何把它与scikit-learn进行比较呢？
OG：通过制作的两个Spark教程，我了解了一下Spark（教程1，教程2）。Spark和Python或scikit-learn之间的主要区别是，Spark默认是一个系统，以分布式的方式管理那些其它数据处理方法无法在内存中处理的数据。这也是MLlib一开始的设计方向（ed：Spark分布式机器学习框架）。他们选择仅实现可扩展性的算法，这些算法可以在它们有能力处理的那些数据上和大量集群中运行。通过只选择有这种特性的算法，他们目前已经解决了这个双重可扩展性问题。
scikit-learn最初的目的是处理内存中的数据，所以我们不存在偏见。我们有一些非常有效的算法，它们只在小数据集上有效。但事实上，我们有很多算法都是以批处理模式实现的。目前，我正在对它们进行重构，主要是为了让其具有更好的可扩展性。
scikit-learn并不是创建跨集群的功能。我们不想改变所有的功能，来处理存储在集群中的资源，但我们想把它作为一种可能性，确保scikit-learn模型可以嵌入到一个类似Spark的框架里，这样它们就可以分布在集群中。例如，当你在训练一个随机森林时，如果你认为你的数据小到可以在整个集群中进行复制，那么你可以很容易地训练每棵树。对于中等规模的数据集，我们也想要加快超参数搜索和交叉验证的速度，这自然就是并行。
在解决集群的分布式计算之前（正如Spark关注的），我对于研究有效的核外处理方法（像Dato正在做的）也是很有兴趣的。目前我还没有真正地研究过细节，但似乎只要你能够更好地进行核外处理并重视算法效率，你就可以减少资源的浪费。这也可能成为scikit-learn未来发展的驱动力。
FD：以分布式方式存储大量数据会导致性能和结果的偏差么？我正在思考使用Spark运行随机森林的例子。
OG：MLlib随机森林算法在选择特征进行划分时，它是直接在每棵树的训练层面进行并行的。它并没有考虑所有可能的分裂。它建立的是一个直方图，并在划分的数据集上进行并行运算。然后，使用总的信息构建划分。这跟估计算法类似。尽管这种方法是近似估算，但在实际应用中，当你使用样本进行建模时，几乎不会出现问题。因为和非估计算法的结果相比非常接近，只是实现的效率差了点。

未来的方向是特征生成?

FD：当你去查看一个数据项目，很多时间 - 如果不是大部分时间 - 是用在数据预处理和特征生成。在过去的几个月里，scikit-learn在朝着特征工程方向发展。这是你将继续维持的方向吗？你会朝一个集成的管道工作吗？这似乎像是一条无止尽的路。有没有一些平行的项目专攻特定的数据类型和格式，同时又遵循scikit-learn的习惯和理念？

OG：在创建scikit-learn预测模型时，特征始终是一个关键点。因为pandas数据框的最新版本，我们越来越善于整合工具箱去操纵任何格式的数据，并把它转为其它格式或是任何其他的表示。
我赞同你的观点，特征工程对于一个具体的应用程序而言，永远是一个特殊环节。我们希望保留一个通用库。如果我们要专攻某个特定的领域并开发特征，它将成为一个独立的特定库的一部分。例如，在天体物理学中有一个叫AstroML的专用库。此前，我在INRIA的团队处理的是影像数据。我们已经开发了一个特定的库，叫做nilearn，它是scikit-learn的一个分支项目。事实上，划分不同项目的范围是很有好处的。它可以围绕社区特定的实践活动进行更好地交流。
FD：在特征工程这个主题上，你相信Spark和MLlib会改变数据科学家的工作方式么？
OG：最近的数据框API是Spark的一个优点。它给了数据科学家一个非常直观，灵活，并富有表现力的工具，用于测试他们不同的数据表示。
从更高层面来讲，最新版本的spark.ml包，允许在以数据组合为特征的“链”中创建管道和预测模型。在链的不同阶段可以交叉验证参数的相互作用。也正是这类API的优点，使它更易于测试。其实在scikit-learn中也可以安装插件，使用数据框作为输入并且添加用户自定义的scikit-learn转换脚本。事实上，使这个过程变得更加简单也正是我们应该努力的实践方向。

搜寻这些项目

FD：非常感谢您这次精彩的谈话！你觉得还有其他任何需要补充的吗？
OG：我认为Python生态圈越来越意识到当前的技术形势，特别是在谈及到处理大量数据时。Java和Scala领先于我们，尤其是Hadoop和Spark。开发人员对于这一点都非常清楚，他们正在寻找答案。如今有很多有趣的项目，如Blaze,Dask,或XRay。他们正在开发相关的APIs，努力使其像Pandas一样出色，并且能做核外计算，最终形成分布式的。Wes McKinney给Cloudera做的Ibis项目也很有趣。它使用的是Python，但用Impala作为后台，用其替代PySpark。其实，我并不相信在当今的生产中能够使用它，但我相信这个主题的发展将会很有趣。
敬请期待Olivier访谈的第二部分，在那里他给数据科学的入门人士和想知道应该培养什么技术的学者提出了一些建议。
原文链接：[Interview] Olivier Grisel on scikitlearn and the future of Machine Learning
technologies (part 1)（编译/刘帝伟 审校/朱正贵、赵屹华 责编/周建丁）

译者简介：刘帝伟，中南大学软件学院在读研究生，关注机器学习、数据挖掘及生物信息领域。

使用**scikit-learn**解释随机森林算法

在以前的一篇博文里，我讨论过如何将随机森林算法转化为一个"白盒"，这样每次预测就能被分解为各项特征的贡献和，即

$$prediction = bias + feature_1 contribution + \ldots + feature_n contribution.$$

我多次想找相关的代码。然而，绝大多数的随机森林算法库（包括scikit-learn）不暴露预测过程的树路径（tree paths）。sklearn的实现方法需要一个额外补丁来暴露。庆幸的是，scikit-learn自0.17版起在API中添加了两项功能，使得这个过程相对而言比较容易理解：获取用于预测的所有叶子节点的ID，并存储所有决策树的所有节点的中间值，而不仅仅只存叶子节点的。结合这两步，就可以获取每次独立预测的预测路径，同时根据查看路径来分解预测过程。
代码已经放在github上了，也可以用 pip install treeinterpreter进行安装。
注意：需要用到仍在开发中的scikit-learn 0.17，你在下面的链接中能找到安装方法 http://scikit-learn.org/stable/install.html#install-bleeding-edge。

用**treeinterpreter**分解随机森林预测

我们选一个简单的数据集，训练一个随机森林模型，并用测试集进行预测，然后分解预测过程。

```
from treeinterpreter import treeinterpreter as tifrom sklearn.tree import
DecisionTreeRegressorfrom sklearn.ensemble import RandomForestRegressorimport numpy
as np from sklearn.datasets import load_bostonboston = load_boston()rf =
RandomForestRegressor()rf.fit(boston.data[:300], boston.target[:300])
```

我们随机挑选两个预测价格不相同的样本。

```
instances = boston.data[[300, 309]]print "Instance 0 prediction:",
rf.predict(instances[0])print "Instance 1 prediction:", rf.predict(instances[1])
```

```
Instance 0 prediction: [ 30.76]
Instance 1 prediction: [ 22.41]
```

随机森林模型对它们的预测结果迥然不同。这是为什么呢？我们接下来就把预测结果分为偏置项（也就是训练集的平均结果）和单个特征贡献值，以便于观察究竟哪些特征项造成了差异，差异程度有多大。
我们直接调用tree interpreter的predict方法，向其传入模型和数据作为参数。

```
prediction, bias, contributions = ti.predict(rf, instances)
```

打印出这些结果：

```
for i in range(len(instances)):  print "Instance", i  print "Bias (trainset mean)", biases[i]
 print "Feature contributions:"  for c, feature in sorted(zip(contributions[i],
 boston.feature_names),                key=lambda x: -abs(x[0])):     print feature, round(c, 2)
 print "-"*20
```

```
Instance 0
Bias (trainset mean) 25.2849333333
Feature contributions:
RM 2.73
LSTAT 1.71
PTRATIO 1.27
ZN 1.04
DIS -0.7
B -0.39
TAX -0.19
CRIM -0.13
RAD 0.11
INDUS 0.06
AGE -0.02
NOX -0.01
CHAS 0.0 --------------------
Instance 1
Bias (trainset mean) 25.2849333333
Feature contributions:
RM -4.88
LSTAT 2.38
```

DIS 0.32
AGE -0.28
TAX -0.23
CRIM 0.16
PTRATIO 0.15
B -0.15
INDUS -0.14
CHAS -0.1
ZN -0.05
NOX -0.05
RAD -0.02
特征贡献值按照其绝对值从大到小排序。我们观察到第一个样本的预测结果较高，正贡献值主要来自RM、LSTAT和PTRATIO特征。第二个样本的预测值则低得多，因为RM特征实际上有很大的负面影响，它不会被其它特征的正面影响所抵消，因此使得预测值要低于数据集的平均水平。
分解的结果真的对吗？很容易检验：偏置和特征贡献值相加应该等于预测值：

print predictionprint biases + np.sum(contributions, axis=1)

[30.76 22.41]
[30.76 22.41]

注意，在把贡献值相加时，我们需要对浮点数进行处理，所以经过四舍五入处理后的值可能略有不同。

比较两个数据集

这个方法的用武之地之一就是比较两个数据集。例如:

- 理解造成两个数据集预测值差异的真正原因，比如是什么因素导致相邻两幢房屋的预测价值差异。
- 调试模型和数据，例如解释为什么新数据的平均预测值和旧数据的不一样。

还是上面这个例子，我们把房价数据的测试集再一分为二，分别计算它们的平均预测价值。

ds1 = boston.data[300:400]ds2 = boston.data[400:] print np.mean(rf.predict(ds1))print np.mean(rf.predict(ds2))

22.1912
18.4773584906

我们发现两个数据集的平均预测价值完全不同。现在我们就能细分导致差异的因素：究竟哪些特征项造成了差异，差异程度有多大。

prediction1, bias1, contributions1 = ti.predict(rf, ds1)prediction2, bias2, contributions2 = ti.predict(rf, ds2)

我们再来计算每一维特征的平均贡献程度。

totalc1 = np.mean(contributions1, axis=0) totalc2 = np.mean(contributions2, axis=0)

由于两个数据集的偏置项都一样（因为模型的训练集都一样），平均预测价值的差异只能来自于特征的贡献值。换句话说，特征贡献差异的总和应该与平均预测的差异相等，我们很容易验证

```
print np.sum(totalc1 - totalc2)print np.mean(prediction1) - np.mean(prediction2)
```

```
3.71384150943
3.71384150943
```

最后，我们把每一维特征贡献的差异之和显示出来，正好就是平均预测值的差异。

```
for c, feature in sorted(zip(totalc1 - totalc2,                boston.feature_names),
reverse=True):  print feature, round(c, 2)
```

```
LSTAT 2.8
CRIM 0.5
RM 0.5
PTRATIO 0.09
AGE 0.08
NOX 0.03
B 0.01
CHAS -0.01
ZN -0.02
RAD -0.03
INDUS -0.03
TAX -0.08
DIS -0.14
```

分类树和森林

同样的方法也能用于分类树，查看特征对某个类别的预测概率值的影响。
我们在iris数据集上做演示。

```
from sklearn.ensemble import RandomForestClassifierfrom sklearn.datasets import
load_irisiris = load_iris() rf = RandomForestClassifier(max_depth = 4)idx =
range(len(iris.target))np.random.shuffle(idx) rf.fit(iris.data[idx][:100], iris.target[idx][:100])
```

接着用一个独立样本做预测。

```
instance = iris.data[idx][100:101]print rf.predict_proba(instance)
```

拆分每一维特征的贡献值：

```
prediction, bias, contributions = ti.predict(rf, instance)print "Prediction", predictionprint "Bias
(trainset prior)", biasprint "Feature contributions:"for c, feature in zip(contributions[0],
      iris.feature_names):  print feature, c
```

```
Prediction [[ 0. 0.9 0.1]]
Bias (trainset prior) [[ 0.36 0.262 0.378]]
```

Feature contributions:
sepal length (cm) [-0.1228614 0.07971035 0.04315104]
sepal width (cm) [0. -0.01352012 0.01352012]
petal length (cm) [-0.11716058 0.24709886 -0.12993828]
petal width (cm) [-0.11997802 0.32471091 -0.20473289]

我们看到对第二类预测能力最强的特征是花瓣长度和宽度，它们极大提高了预测的概率值。

总结

让随机森林算法的预测结果具有解释性也很容易，几乎达到了线性模型的解释能力。有了treeinterpreter，这个步骤只需几行代码就能搞定。
原文地址：**Random forest interpretation with scikit-learn**（译者/赵屹华 校检/刘帝伟、朱正贵、李子健 责编/周建丁）
赵屹华，计算广告工程师@搜狗，前生物医学工程师，关注推荐算法、机器学习领域。

文章原链接：http://www.csdn.net/article/2015-10-08/2825851

上一篇：[转载]基于Hadoop集群的大规模分布式深度学习
下一篇：[转载]英特尔院士Pradeep Dubey概述深度学习愿景与优化